NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

KNOWLEDGE QUALITY FUNCTIONS
FOR RULE DISCOVERY

by

Elizabeth S. Walters
and
Frank J. Bunn

September 1994

Thesis Advisor: Balasubramaniam Ramesh

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE KNOWLEDGE QUALITY FUNCTIONS FOR RULE DISCOVERY	5. FUNDING NUMBERS
6. AUTHOR(S) Elizabeth S. Walters and Frank J. Bunn	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT *(maximum 200 words)*

The Department of Defense (DoD) possesses tremendous amounts of data stored in many large databases. Due to the size of these databases, humans are incapable of efficiently discovering interesting and useful patterns so an automated data-mining tool is necessary. Output in the form of production rules, *i.e.*, "If *y* Then *x*," is preferred because they are understandable by humans and support decision making processes.

This thesis investigates the manner in which data-mining systems discover useful, interesting, but currently unavailable knowledge. The search and evaluation process, guided by a knowledge quality function, is the key task of a data-mining system.

This thesis evaluates three knowledge quality functions taken from the literature. Each knowledge quality function discovers new and interesting sets of rules reflecting different characteristics of knowledge. DoD applications are suggested for each of the knowledge quality functions.

14. SUBJECT TERMS Knowledge Discovery, Data-mining, Fitness, Quality Functions	15. NUMBER OF PAGES 63
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Approved for public release; distribution is unlimited.

KNOWLEDGE QUALITY FUNCTIONS
FOR RULE DISCOVERY

by

Elizabeth S. Walters
Lieutenant Commander, United States Navy
A. B., Whitman College, 1976

Frank J. Bunn
Lieutenant Commander, United States Navy
B.S., Auburn University, 1980

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1994

Authors: Elizabeth S. Walters

Frank J. Bunn

Approved by: Balasubramaniam Ramesh, Thesis Advisor

William B. Short, Thesis Associate Advisor

David R. Whipple, Chairman, Department of Systems Management

ABSTRACT

The Department of Defense (DoD) possesses tremendous amounts of data stored in many large databases. Due to the size of these databases, humans are incapable of efficiently discovering interesting and useful patterns so an automated data-mining tool is necessary. Output in the form of production rules, i.e., "If y Then x," is preferred because they are understandable by humans and support decision making processes.

This thesis investigates the manner in which data-mining systems discover useful, interesting, but currently unavailable knowledge. The search and evaluation process, guided by a knowledge quality function, is the key task of a data-mining system.

This thesis evaluates three knowledge quality functions taken from the literature. Each knowledge quality function discovers new and interesting sets of rules reflecting different characteristics of knowledge. DoD applications are suggested for each of the knowledge quality functions.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

John Walters did an excellent job of editing all the chapters. His help, advice, and encouragement is much appreciated.

We thank John, Lyn, Andrew and Michael for their wonderful patience and support.

I. INTRODUCTION

A. GENERAL DESCRIPTION OF PROBLEM

The Department of Defense (DoD) possesses tremendous amounts of data stored in databases: financial information, personnel records, material consumption, transportation requirements, fuel consumption, flying-hour costs, pharmaceutical usage, aviation safety incidents, material deficiencies, material casualties, enemy submarine sonar signatures, travel expenses, etc. Currently, these data are accessed to produce reports, statistics and answer queries. Managers in many organizations finding themselves in the possession of large and rapidly growing databases are beginning to suspect the information in their databases is not used to the fullest potential. For example, if a suitable "data mining" tool were applied to aviation maintenance data, we might discover that a particular avionics component unexpectedly develops an unusually high failure rate--but only in those aircraft in which a new type of fuse has been introduced. Humans are unlikely to discover any but the most obvious and uninteresting patterns in the data. Revolutionary improvements could be made if the underlying patterns of behavior--hidden in our databases--were understood better in the areas of intelligence, manufacturing process control, procurement, inventory management, etc.

B. RESEARCH OBJECTIVE

The primary objective of this thesis is to test the performance of several knowledge quality functions using the Naval Postgraduate School Genetic Program (NPSGP), a

knowledge discovery tool. In most data-mining systems the search is guided by an evaluation function, in our case the knowledge quality function. In this thesis we attempt to show how the knowledge quality function can be constructed to reflect specific characteristics of knowledge which are likely to be of importance to users of data-mining systems.

C. RESEARCH METHODOLOGY

In this thesis, we use NPSGP, a prototype data mining system, to study the data mining performance of three knowledge quality measurement functions found in the literature. We present the set of best rules discovered from a large database by each of the three quality functions tested and evaluate the utility of the these functions in the context of data mining scenarios.

Based on our evaluations, numerous changes have been made to the NPSGP. Hundreds of hours were spent testing many configurations of program parameters, how to prepare various types of data for use in NPSGP, learning how to interpret the output, and verifying the output's validity. We developed protocols for executing NPSGP and methods for documenting, tracking and interpreting the results. Using four databases held in the University of California at Irvine's Repository of Machine Learning Databases and Domain Theories, we tested NPSGP's ability to handle different types of data: all continuous data, all discrete data, and a mix of continuous and discrete data.

D. ORGANIZATION OF STUDY

Chapter I provides general background for this study. Chapter II discusses the mechanics of data mining systems: learning strategies, typical applications, types of

knowledge representation, and search techniques. Chapter III discusses the problem of deriving knowledge from data and surveys several proposed measures of knowledge quality content. Chapter IV details how to use NPSGP as a data mining system. Chapter V compares the results of three different knowledge quality functions used in NPSGP. Chapter VI presents conclusions and recommendations for continuing this line of research.

II. DATA MINING

Widespread use of computers in industry and government creates mountains of raw data. Buried within these mountains of data are patterns of potentially great value to the Department of Defense:

- Patterns of adversary submarine and aircraft behavior
- Patterns of demand for material stocked in the military supply system
- Patterns of material failure in equipment and repair parts.

Methods are needed to extract this valuable high-level information (knowledge). The use of automated systems to find new knowledge is necessary and worthwhile because no organization can afford the cost of manually examining and analyzing the typical large corporate database in this pursuit (Smyth and Goodman, 1991, p. 160). Data mining is the specialized area of machine learning that uses computers to extract knowledge from databases.

A. DATA-MINING SYSTEMS

The nature of real world databases presents challenges to the data miner:

- Databases are rarely designed with data mining in mind.
- Databases commonly noisy, containing erroneous or missing data.
- Databases often represent only a small subset of the true population about which knowledge is desired.
- Databases are typically very large and constantly changing.
- Databases often represent behaviors which can not be modeled mathematically.

Data mining systems must be robust enough to deal with these challenges and still produce useful, interesting knowledge.

Data-mining systems are primarily concerned with three knowledge problems: classification, association, and sequencing. Classification involves partitioning objects in the database into groups. Actual applications of classification systems include credit approval and treatment-appropriateness determination. The problem of association involves generalizing on interesting patterns discovered in the database. The system attempts to discover and describe knowledge about a specified (target) database field or attribute in the terms of other (non-target) attributes. Usually, the user is interested in sets of rules satisfying some specification. Actual applications of association systems include detection of faults in a manufacturing process and modeling consumer behavior. Sequencing involves finding connections among temporally ordered data (Agrawal *et al.*, 1993, p. 915). Figure 2-1 graphically presents this process of knowledge discovery.

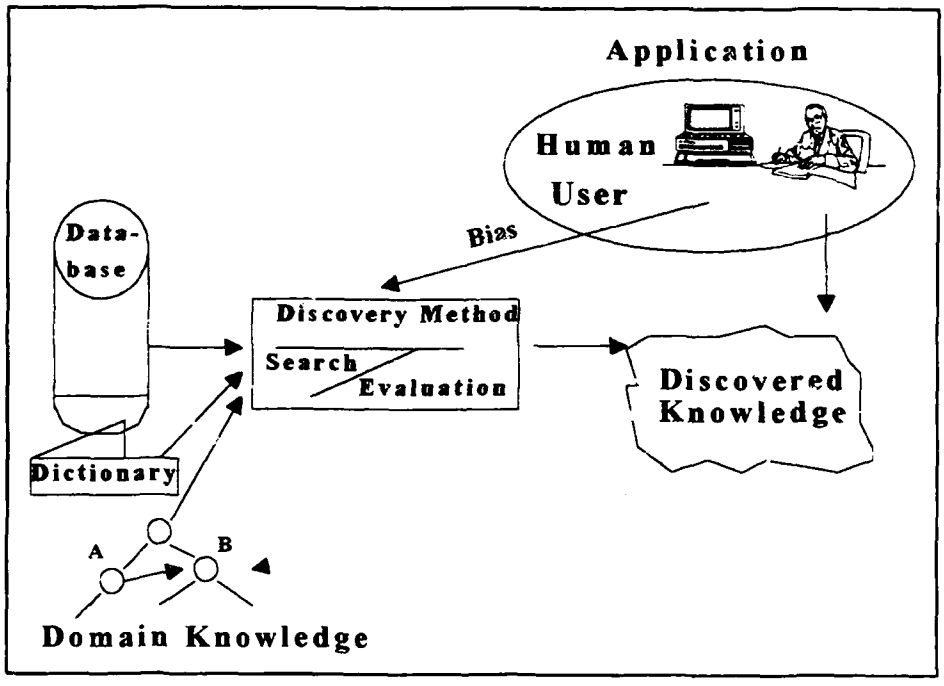


Figure 2-1. A Framework for Knowledge Discovery in Databases
(Adapted from Frawley *et al.*, 1991, p. 61)

Actual applications of sequencing include modeling stock market movements and weather forecasting. In all cases, the data mining task is to find the patterns with the greatest utility to the user. Our research addresses only the problem of association. Hereafter, the term "data-mining system" is used to refer to an "associative system."

B. KNOWLEDGE REPRESENTATION

Discovered knowledge is represented in most data mining systems by one of two methods: production rules or decision trees. Other representational formats such as neural networks, semantic nets, and decision lists are occasionally used by data mining systems but are beyond the scope of this paper.

1. Decision Trees

Decision trees provide a map of the relations found among the data. Ordinarily, nodes of decision trees are labeled with attribute names, the edges are labeled with possible values for this attribute, and the leaves are labeled with the different classes of the target attribute. A class is described by the path of nodes and leaves which lead to it (Holsheimer, 1994 p. 42). Figure 2-2 illustrates a typical decision tree.

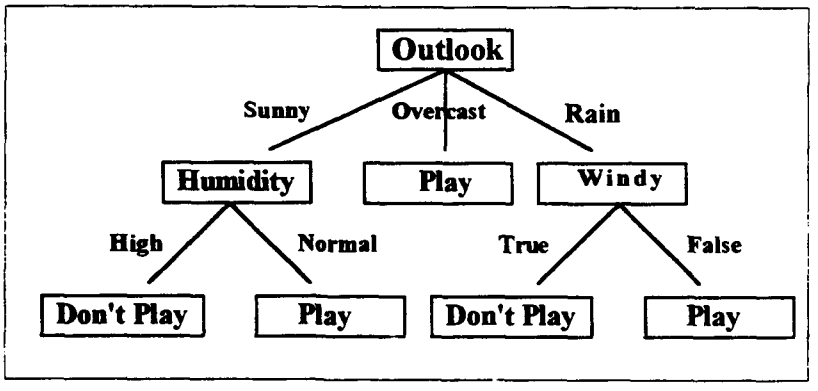


Figure 2-2. Decision Tree (Holsheimer, 1994, p. 42)

Systems using decision trees are essentially sequential decision algorithms. Trees must always begin with the attribute associated with the root node and partition the data into branches based on values of attributes. Systems which use decision tree representations are not designed to accommodate missing values (Smyth and Goodman, 1992, p. 303). Decision trees "tend to grow very large for realistic applications and are thus difficult to interpret by humans" (Holsheimer, 1994, p. 42). Decision trees also grow excessively complicated in the presence of noisy databases (Dhar and Tuzhilin, 1993, p. 930). Most systems that use decision-tree representations implement a pruning mechanism to offset this tendency to overfit noisy data (Quinlan, 1986, p. 154). Decision trees may be appropriate if the reasoning process is complex and it is not necessary to understand the underlying data relationships in order for the results to be useful. Decision trees are also at an advantage when the results of the data mining system will be directly input into other computer programs (Frawley *et al.*, 1991, p. 65).

2. Production Rules

Production rules represent relationships between attributes. Production rules used by data mining systems appear in the form: "If *description y* Then *target attribute class x*" where y is in terms of the non-target attributes.[1] A degree of certainty or confidence (the probability of *x* given *y*) is usually associated with production rules. Production rules have the advantage of being familiar and easily understood by humans (Holsheimer, 1994 ibid.). For example, the knowledge built into expert systems frequently takes the form of production rules. In some respects, the rules generated by data mining

[1] Note: this notation convention is the reverse of the traditional: If *x* Then *y*.

systems can be understood and used as machine generated expertise (Smyth and Goodman, 1991, p. 168). Because of their inherent clarity, production rules are most appropriate in decision support systems where human understanding of the underlying relationships between the attributes is necessary to take appropriate action.

Systems using production rules are data driven in the sense that any set of input data can potentially be used to begin the inference. In addition, rule-based systems can accommodate missing attribute information. In general, rule-based systems are more flexible than systems using decision tree structures (Smyth and Goodman, 1992, p. 303). The use that will be made of the knowledge found by data mining systems should determine the way the results are represented. The form the representation takes quite often drives the logic by which the knowledge is derived. Whatever representation the knowledge takes--decision trees, "If...Then" rules, etc.--users must remember that only descriptions of relationships are expressed; the conditions necessary to support causation may not be present.

C. DATA MINING SEARCH TECHNIQUES

The primary task of a data-mining system is to search for general patterns that describe the classes of the designated attribute in terms of the other attributes. If each non-target attribute has the same number of discrete states, the maximum number of possible descriptions for each target attribute class can be calculated as:

$$m\frac{m!}{(m-r)!} \qquad (2\text{-}1)$$

where m is the number of discrete states and r is the number of attributes (Weiss and Hassett, 1991, p. 217). Table 2-1 shows that the possible number of descriptions for all

classes in the target attribute increases geometrically with the number of attributes or discrete states. If attributes are represented by non-discrete (continuous) data, the number of possible descriptions expands without limit. In non-trivial-sized databases, it is normal

TABLE 2-1. NUMBER OF POSSIBLE DESCRIPTIONS

Number of discrete states	Number of Attributes			
	5	10	15	20
2	40	180	420	760
4	480	20,160	131,040	465,120
6	4,320	907,200	21,621,600	167,443,200

for all instances to represent only a small proportion of the potential descriptions. Evaluating every existing description in the database is the only way to ensure the best set of patterns is found. This strategy has the disadvantage of generating too many patterns, many of which are obvious, redundant, or useless (Piatetsky-Shapiro, 1993, p. 5). An exhaustive search also leads to slow processing (Holshimer, 1994, p. 33). The actual application must determine whether speed or solution optimality should be the priority. For example, in a rapidly evolving battlefield situation, a quick response is more important than finding the optimal solution. On the other hand, if the objective of the data mining system is to support staff work, (*e.g.*, model the behavior of salaried military physicians in order to design an incentive system to encourage seeing more patients), time is less critical than the optimality of the solution. To cope with these challenges, data mining search strategies are often guided by statistically-based criteria such as the quality functions to be discussed in the next chapter. Several types of search techniques have been developed to

find high-quality descriptions about the target attribute classes while avoiding exhaustive searches of the description space.

1. Traditional Techniques

The two general systematic search techniques used to find the best descriptions attempt to simulate human reasoning processes and are called "bottom-up" and "top-down". In "bottom-up" or data-driven techniques, the set of initial descriptions for a given class consists of all examples in the database for that class. Commonality among the attributes is sought across the initial set of descriptions. Where commonality is found, new, more useful descriptions of the target class are created. In "top-down" or model-driven techniques, the initial set of descriptions consists of the most general rules possible. Both specialization and generalization operations are then used to produce new, more useful descriptions. These systematic techniques are implemented either by "irrevocable" or "tentative" search strategies. Irrevocable search strategies apply a selected operation until a terminal result is achieved. This can be viewed as pursuing a hierarchical path that precludes reconsideration of an alternate path once it has been rejected. Tentative search strategies allow backtracking and therefore may achieve an improved description, where an irrevocable path might settle on a local maximum. The tentative strategy is more flexible but requires more computer memory (Holsheimer, 1994 p. 31). Almost all data mining systems appearing in the knowledge discovery literature use the traditional "top-down" or "bottom-up" techniques.

2. Genetic Based Learning Techniques

If the search problem can be seen as an optimization problem, systems using genetic algorithms and genetic programming have been found to be effective at optimizing on computationally based functions. Introduced by Holland in the early 1960s, genetic algorithms (GA) imitate the mechanics of biological natural selection. Genetic algorithms use operations analogous to crossover and mutation in cell division to propagate modifications of descriptions across iterations (generations). This process is illustrated in Figure 2-3.

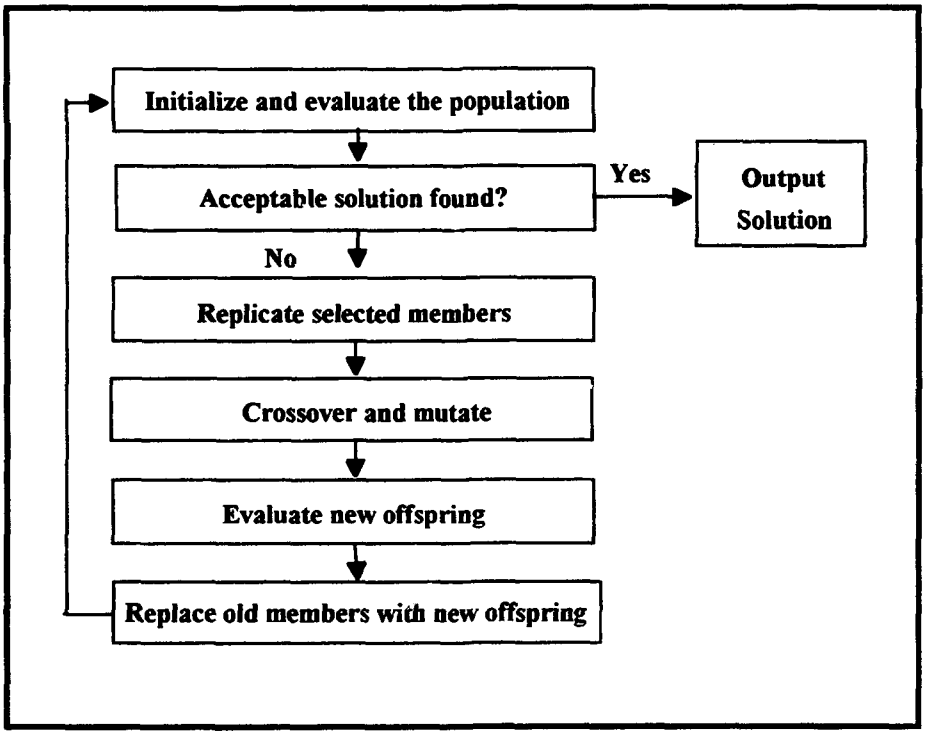


Figure 2-3. Paradigm of a Genetic Algorithm (Grefenstette, 1993, p. 6)

In the context of data mining, a descriptive statement in a genetic algorithm appears as a fixed length string of ones and zeroes, where each position in the string

corresponds with one attribute (*i.e.*, field of the database). A large number of strings are generated at random and evaluated for fitness. The quality of the strings, as measured by a quality (fitness) function, determines which strings will participate in the crossover and mutation operations.

> In Holland's scheme, crossover allowed important building blocks of high fitness to carry over into the next generation. These formed a base from which the genome [the set of characteristics encoded as genes for each species] could more successfully evolve. As these blocks met up with other successful building blocks, the result could be new and innovative approaches to the difficulties offered by the environment. Thus the process delivered what Holland thought of as evolution's greatest virtue: its perpetual novelty in its approaches to maintaining fitness. (Levy, 1992, p. 169)

Genetic algorithms generate high quality descriptions but have less tendency to terminate on local optima than traditional techniques. Genetic algorithms "outperform traditional learning techniques, especially when the descriptions that have to be learned are complex...or when no domain knowledge is available," (Holsheimer, 1994 p. 34) or when the database is noisy (Goldberg, 1994, p. 114).

Building on the base of genetic algorithms, Koza introduced genetic programming (GP) in the late 1980's.

> [Koza's] breakthrough was deciding to identify the units of crossover not as single characters, or even as lines in a computer program, but as symbolic expressions (S-expressions) written in the LISP syntax. Made of mathematical functions and inputs appropriate to the problem, these S-expressions were essentially subroutines, which were commonly viewed as tree structures. These subroutines could be successfully crossed over so that in a reasonable percentage of matings, the offspring computer program would conform to syntax at least as well as its parents did. Another way of viewing it was that the S-expressions formed tree-shaped "chromosomes." Crossover was the equivalent of swapping branches. (Levy, 1992, p. 176)

While genetic programming has been used successfully in the areas of robotics, game-playing, and discovering mathematical theorems, its value as a data mining technique has yet to be thoroughly tested. Later in this paper, we describe a prototype genetic programming system used for data mining. In this system, each "rule" is treated as a program. The entire parent rule or parts of it can be paired with another parent rule or rule fragment to produce offspring rules. The programs retain the positive attributes of rule-based systems and avoid the negative aspects of the decision trees discussed previously.

D. SUMMARY

The complexity of the task facing the data miner requires robust systems that can produce useful and interesting knowledge. Ideally, a knowledge discovery system:

- can deal with very large databases (potentially of terabyte size)
- can deal with continuous as well as discrete data (Smyth and Goodman, 1992, p. 301)
- can deal with noisy datasets
- searches throughout the search space for general patterns in the data
- formulates a statement (individual piece of knowledge)
- evaluates and orders each statement according to the user's criteria for usefulness
- determines if the statement should be retained, modified or rejected in the context of the other statements (Piatetsky-Shapiro *et al.*, 1993, p. 5)
- presents the knowledge to the user in an understandable format
- provides results in a time-frame that is satisfactory to the user
- has a well-designed user interface.

Effective data mining system design must consider which search technique, guidance mechanism, and knowledge representation will most appropriately produce the knowledge needed by the user.

III. KNOWLEDGE DISCOVERY

Knowledge discovery in databases, also known as data-mining, is one of the fastest growing areas within the field of artificial intelligence. Knowledge discovery has been defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" (Frawley *et al.*, 1991, p. 3). Zytkow defines *knowledge discovery* as the "acquisition of objective knowledge" as distinct from *learning*, which is defined as acquiring knowledge that is already known. Therefore, discovery must precede learning; once a piece of knowledge is discovered, it can be learned (Zytkow, 1993, p. 7). In this chapter we review how knowledge is inferred, characterized and quantified. Finally, we present and analyze several proposed measures of knowledge quality.

A. INFERENCE

We differentiate between *data* and *knowledge*. *Data* is defined as the facts upon which a reasoning process is based; *knowledge* is defined as the logical conclusion of a reasoning process. Databases are collections of facts about objects found in a common environment. Two basic processes are used to infer knowledge from raw data: deduction and induction.

1. Deduction

Deduction is the process of reasoning from the general to the particular; that is, rationally drawing specific conclusions from more general principles which are assumed to be true. By Zytkow's definition, deduction is "learning." Deduction allows inference of

specific knowledge about relationships between data elements in the database, much as a syllogism is constructed. Statistical knowledge such as averages, ranges, distributions, etc., can also be produced based on deductive reasoning. Traditional expert systems attached to databases are good examples of systems based on deductive reasoning. The human expert knowledge base is assumed to be true. The inference engine draws on the specific knowledge contained in the database to deduce knowledge. Deductively derived knowledge is provably correct if the data provided and the general principles are correct.

For example, the following is an excerpt from a report prepared by DRAIR ADVISER, an expert system used by the U.S. Air Force at Tinker Air Force Base, in response to a request about the performance of a particular high-cost aviation component:

> MAINTENANCE DATA (D056): A total of 175 inherent failures occurred between JUL 1991 and JUN 1992, which translates into a Mean Time Between Maintenance Type-1 (MTBF-1) of 162 hours. There were no aborts reported. The MTBM-1 trend shows a decrease of 4.9 hours per month. A total of 332 maintenance actions resulted in a MTBM-Total of 85 hours. The percentage of inherent failures to total maintenance action is 52.7%. The retest OK rate (42%) exceeds 8% (Robey *et al.*, 1994, p. 68).

The expert system uses the definition of Mean Time Between Failures (MTBF) and actual data in the 1.6 GB database to calculate an actual MTBF for a specific component (Robey *et al.*, 1994, p. 69). This is a deductive reasoning process.

2. Induction

Induction is the process of reasoning from the particular to the general; that is, drawing conclusions based on generalized patterns found in the facts. By Zytkow's definition, induction is "discovery." Data mining is the process of applying inductive

reasoning to databases of facts. Each pattern is a piece of knowledge, and taken as a whole these patterns create a model of the database. Knowledge produced by an inductive system may be consistent with the environment from which the database was drawn; that knowledge however is not necessarily logically provable in the same way as deductive knowledge (Yasdi, 1991, p. 298). Because the process of inference by induction does not require prior knowledge, it is more independent of the user than deductive systems and therefore more likely to discover knowledge previously unknown to the user, which is the essence of data mining.

The induction process can be demonstrated by searching for general patterns in Table 3-1, a hypothetical database of defective F-14 repair parts.

TABLE 3-1 SAMPLE F-14 REPAIR PARTS DATABASE

Part Name	Airframe	Defect	Manufacturer
Flight Computer	F14-A	Seal Broken	ABC
Altimeter	F14-C	Seal Broken	XYZ
Heads-up Display	F14-A	Cracked	XYZ
Fairing	F14-C	Seal Broken	ABC
Flight Computer	F14-A	Seal Broken	ABC
Navigation Computer	F14-A	Seal Broken	ABC
Fairing	F14-C	Cracked	XYZ

Several patterns are obvious which can be stated in terms of the attributes of the objects in the database:

- 80% of defective parts with broken seals were manufactured by ABC.
- 100% of defective parts with cracks were manufactured by XYZ.
- 67% of defective parts manufactured by XYZ had cracks.

Given a real-world database of defective aviation parts such as the one used by the U.S. Air Force at Tinker Air Force Base, an inductive data-mining system might have identified an expensive aviation repair part with a high failure rate *but only when used in one model of aircraft.* Inductive systems discover the information that no one knows to ask for, while deductive systems provide data to support the patterns so obvious that someone decided to analyze them.

B. KNOWLEDGE QUALITY

High quality knowledge is readily understandable, has importance in the context of the real world, and most importantly facilitates the goal(s) of the user (Frawley *et al.*, 1991, p. 4). Researchers suggest operationally desirable characteristics of quality knowledge patterns in the context of "If y, then x" rules:

- Past predictive usefulness -- the description (y) is a good predictor of the outcome (x). In probability terms, this is expressed as $p(x|y)$.
- Simplicity of the pattern (Occam's razor)--simpler patterns are more likely to be correct for data not represented in the database. $p(y)$ is useful as a surrogate for simplicity (Smyth and Goodman, 1992, p. 305).
- Novelty -- the pattern is previously unknown to the user (Frawley *et al.*, 1991, p. 4)
- Uniqueness -- the pattern is not redundant (Major and Mangano, 1993, p. 31).
- Complexity -- the pattern cannot be derived through trivial computations (Frawley *et al.*, 1992, p. 59) and is well integrated with other relevant data (Inmon, 1993).
- Statistical significance -- with some degree of certainty, the pattern does not occur by chance.

Not every characteristic listed above is important in every data mining application. In fact, some of these characteristics (*e.g.*, past predictive usefulness and novelty) appear to be in conflict. Therefore only the user can determine which characteristics of knowledge are appropriate to the current application.

Once the user has determined which characteristics are desirable for a given data mining application, a way must be found to determine if the knowledge discovered reflects those desired characteristics. To the extent the desired characteristics can be quantified, a knowledge quality function Q can be formed and used to measure the quality of each piece of knowledge discovered by the data mining system. As discussed in the last chapter, a quality function is often used by data mining systems to guide the search for quality patterns.

C. QUANTIFYING KNOWLEDGE QUALITY

1. Common Terms in Knowledge Quality Functions

Most measures of knowledge quality described in the current knowledge discovery literature are functions of: the probability of a description, $p(y)$; the probability of an outcome, $p(x)$; the probability of an outcome given a description, $p(x|y)$; and description complexity. Relative weights can be introduced to reflect user biases about the desirability of specific characteristics of knowledge quality (Piatetsky-Shapiro, 1991, p. 231). Several quality functions have been proposed by researchers in the data mining field. To simplify the discussion, these functions will be stated in probability-oriented terms:

- X is the attribute which is made up of classes (x).
- x is the specific class within X that is to be described by y, or the right hand side (RHS) of a rule.
- y is the description of class x to be evaluated, or the left hand side (LHS) of a rule.
- n is the total number of examples in the source database.
- $p(y)$ is the probability of the description (LHS) occurring in the database, sometimes used as a surrogate for description simplicity (i.e., descriptions with few attributes are the most likely to occur in a database).

- $p(x)$ is the probability that any fact (example) in the database is in the class x, or the *a priori* value of any rule.
- $p(x|y)$ is the conditional probability that x will occur given description y, or the *a posteriori* value of the rule.

2. Principles of Behavior for Knowledge Quality Functions

Piatetsky-Shapiro proposes three "intuitively correct" principles for behavior of knowledge quality functions (Q):

- If x and y are independent *(i.e.,* $p(x|y) = p(x)$*)* the rule is not interesting, and Q = 0.
- Q monotonically increases when $p(x|y)$ increases and other factors remain equal. This principle indicates a bias toward the characteristic of past predictive usefulness.
- Q monotonically decreases when $p(x)$ increases and other factors remain equal. This principle indicates a bias towards "rare" outcomes, *i.e.*, those that occur less frequently (Piatetsky-Shapiro, 1991, p. 232).

Each of the quality functions discussed below will be evaluated for compliance with the three proposed principles. Piatestsky-Shapiro further conjectures that all measures of rule quality that satisfy these principles will produce the same rules in approximately the same order *(i.e.*, sorted on the quality function) (Piatetsky-Shapiro, 1991, p. 246). This conjecture will be tested later in this thesis.

D. KNOWLEDGE QUALITY FUNCTIONS

1. Information Theory

Information-theoretic approaches to quantifying the quality of knowledge were discussed as early as 1948, when Wiener defined the information content associated with an event as the difference between the knowledge value before the event and knowledge value after the event (Frawley *et al.*, 1991, p. 267). Many knowledge quality functions are based on information theory, where information content is considered

> a measure of the freedom of choice with which a message is selected from the set of all possible messages. The mathematical expression for information content closely

resembles the expression for entropy in thermodynamics. The greater the information in a message, the lower its randomness, or "noisiness," and hence the smaller its entropy. (The Concise Columbia Encyclopedia, 1983, p. 409)

The classic equation for the information content of a single piece of knowledge is :

$$entropy = -p(x|y) \bullet \log_2(p(x|y)) - (1-p(x|y)) \bullet \log_2(1-p(x|y)) \quad (3\text{-}1)$$

(Quinlan, 1986, p. 151). Figure 3-1 shows that entropy does not comply with Piatetsky-Shapiro's three principles for quality function behavior because it:

- is not zero when $p(x|y) = p(x)$,
- does not increase monotonically with $p(x|y)$, and
- does not vary with $p(x)$.

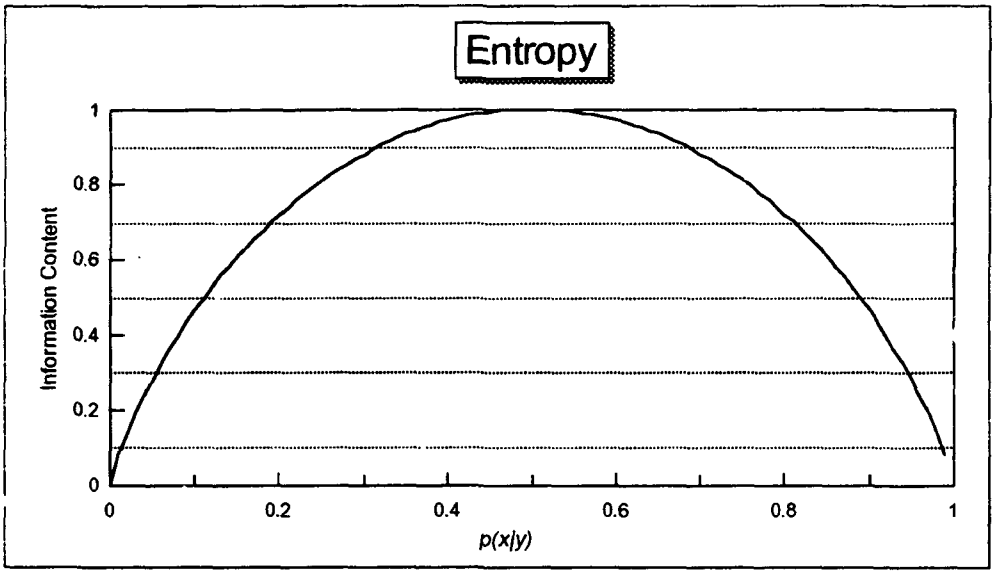


Figure 3-1. Entropy as a Function of $p(x|y)$.

2. Rule Interest and Φ

Piatetsky-Shapiro proposes two knowledge quality functions, Rule Interest and Φ, that satisfy his criteria for the behavior of knowledge quality functions. Rule Interest is offered as the simplest quality function that satisfies his three intuitive principles:

$$Rule\ Interest = [\, p(x|y) - p(x) \,] \bullet p(y)] \quad (3\text{-}2)$$

(Smyth and Goodman, 1991, p. 163). Rule Interest measures the difference between the actual number of instances where both the description (y) and the outcome (x) are true, and the number expected if the outcome (x) were independent of of the description (y). The standard statistical measurement for the significance of the correlation between y and x, Φ, is the other function offered by Piatetsky-Shapiro that satisfies the three principles:

$$\Phi = \frac{(p(x|y) - p(x)) \bullet p(y)}{\sqrt{p(y) \bullet p(x) \bullet [1 - p(x)] \bullet [1 - p(y)]}} \qquad (3\text{-}3)$$

(Piatetsky-Shapiro, 1991, p. 232). Figures 3-2 and 3-3 illustrate how Rule Interest and Φ, respectively, vary with $p(x|y)$ for several values of $p(x)$. Rule Interest and Φ comply with Piatetsky-Shapiro's three principles for quality function behavior. They:

- are zero when $p(x|y) = p(x)$, so rules without information gain have a value of zero;
- increase monotonically with $p(x|y)$; and
- decrease monotonically with $p(x)$.

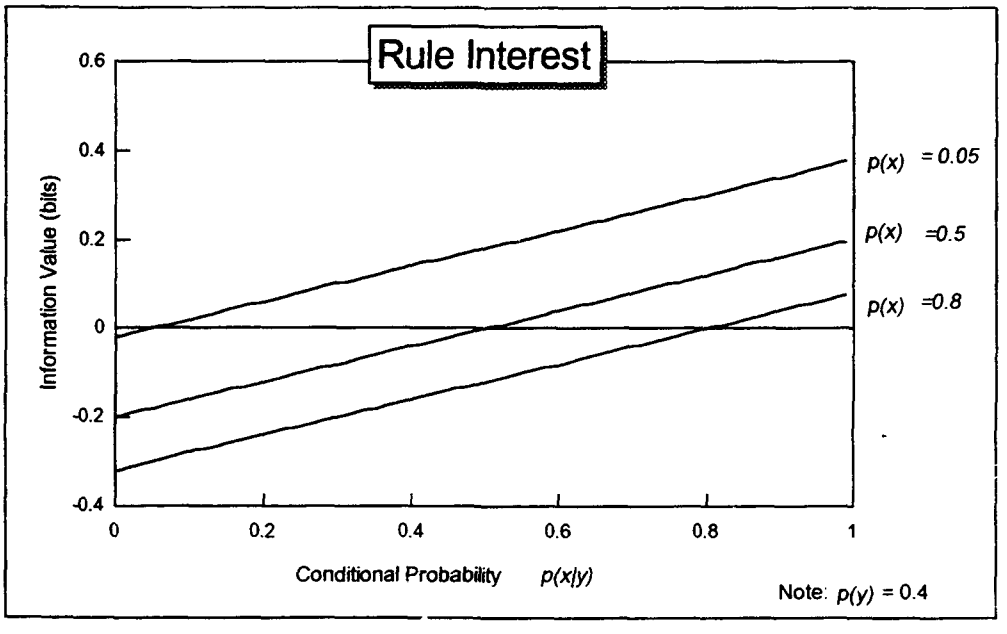


Figure 3-2. Rule Interest for Several Values of $p(x)$.

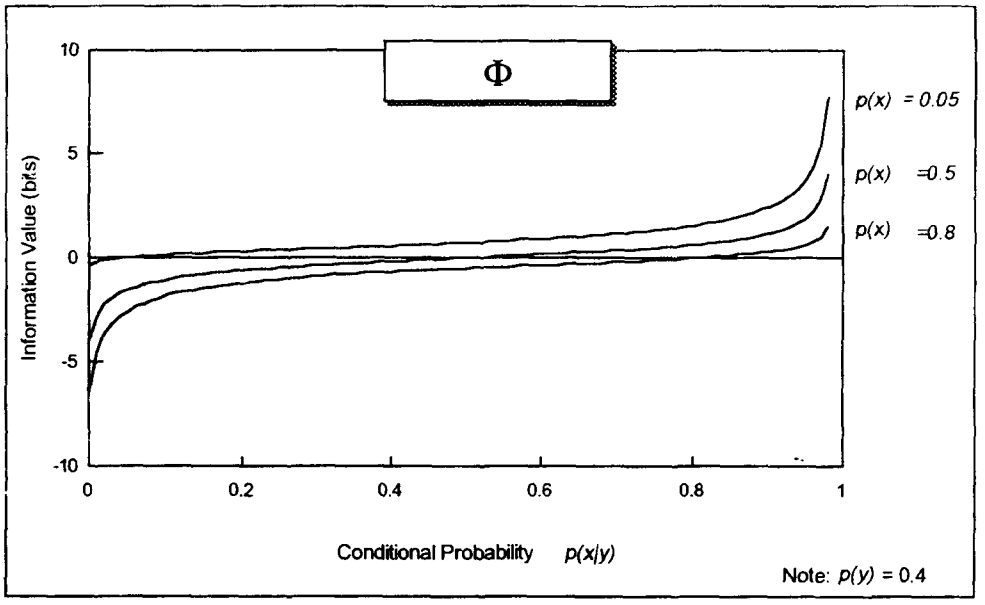


Figure 3-3. Φ for Several Values of $p(x)$.

3. J-measure

Smyth and Goodman strongly support using information theory concepts to measure knowledge quality (Smyth and Goodman, 1992, p. 304). Their function, J-measure, is derived from the classic formula (3-1). Entropy is modified to emphasize the *a priori* level of information, $p(x)$. The result is called *j-measure:*

$$j\text{-}measure = p(x|y) \bullet \log \frac{p(x|y)}{p(x)} + (1 - p(x|y)) \bullet \log \frac{1-p(x|y)}{1-p(x)} \qquad (3\text{-}4)$$

Smyth and Goodman hold that j-measure appears in the information theory literature as "cross-entropy" or "discrimination." J-measure is the product of j-measure and $p(y)$, the surrogate for simplicity.

$$J\text{-}measure = p(y) \bullet j\text{-}measure \qquad \text{or} \qquad (3\text{-}5)$$

$$J\text{-}measure = p(y) \bullet \left[p(x|y) \bullet \log \frac{p(x|y)}{p(x)} + (1 - p(x|y)) \bullet \log \frac{1-p(x|y)}{1-p(x)}\right] \qquad (3\text{-}6)$$

(Smyth and Goodman, 1991, p. 163).

In theory, J-measure emphasizes the rare outcome, low $p(x)$ more than either Rule Interest or Φ. Figure 3-4 illustrates how J-measure behaves with respect to $p(x|y)$ for several values of $p(x)$. J-measure:

- is zero where $p(x|y) = p(x)$, so rules without information gain have a value of zero;
- does not increase monotonically with $p(x|y)$ over the full range of $p(x|y)$, but does so when $p(x|y) > p(x)$;
- does not decrease monotonically with $p(x)$ over the full range of $p(x|y)$, but does so when $p(x|y) > p(x)$.

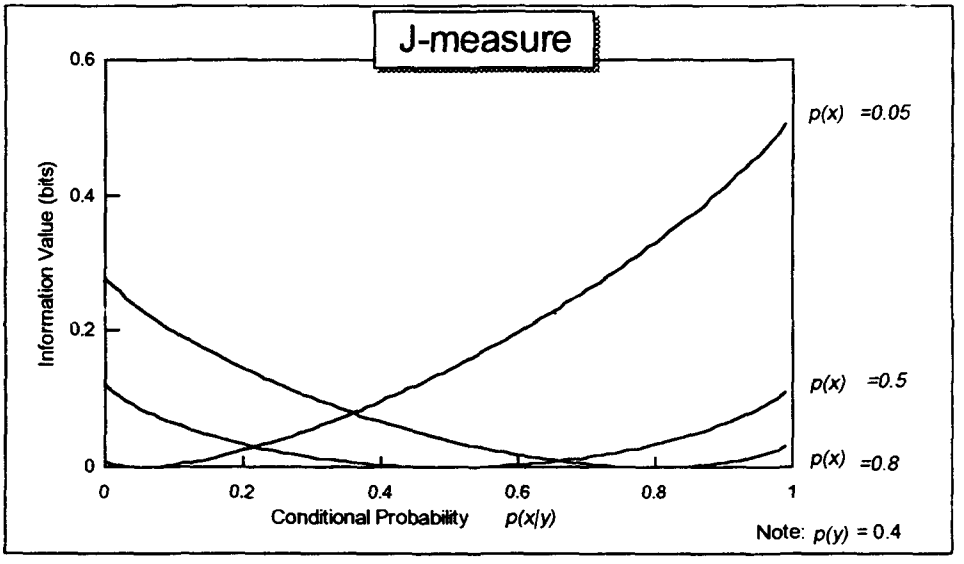


Figure 3-4. J-measure for Several Values of $p(x)$.

4. Other Knowledge Quality Functions

Chan and Wong offer another function W for knowledge quality based on knowledge theory (Chan and Wong, 1991, p. 117):

$$W = \log \frac{p(x|y)}{p(x|not\,y)} \tag{3-7}$$

W is interpreted as "a measure of the difference in the gain in knowledge when an object characterized by *y* is assigned to *x* and when it is assigned to other classes" (Chan and Wong, 1991, p. 112). As can be seen in Figure 3-5, Chan and Wong's *W* function:

- is zero only when $p(x|y) = 0.5$, where $p(x|y) = p(x|\ not\ y)$, and not where $p(x|y) = p(x)$, so that rules without knowledge gain may have positive values;
- increases monotonically with $p(x|y)$, other things being equal;
- does not decrease monotonically with $p(x)$ because it is not a function of $p(x)$, so rules about rare events will probably not be generated.

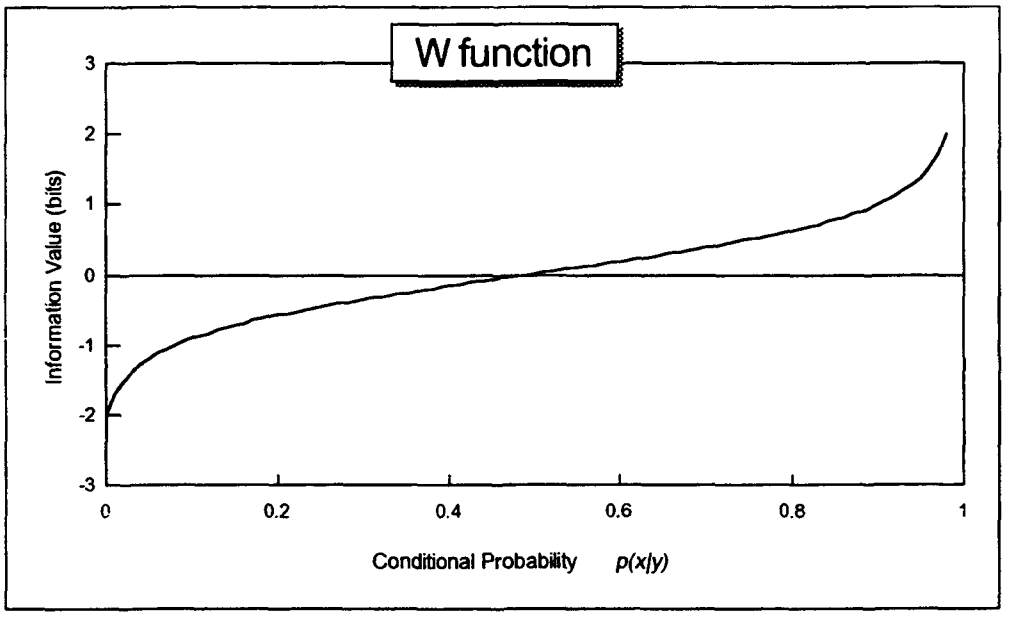


Figure 3-5. Plot of *W* as a Function of $p(x|y)$.

The *W* function satisfies only one of Piatetsky-Shapiro's principles for knowledge quality function behavior and appears to be limited to the specific purpose of "acquiring classificatory knowledge from an imperfect database" (Chan and Wong, 1991, p. 110).

Other measures of knowledge quality not based on information theory have been used by researchers in the knowledge discovery field. An intuitively appealing measure is:

$$Certainty = p(x|y) \tag{3-8}$$

This concept is also frequently called "strength" or "confidence." Certainty does not satisfy the first and third of Piatetsky-Shapiro's principles because it is not a function of $p(x)$. Certainty can meet the first principle if rules with values at or below $p(x)$ are excluded.

Table 3-2 summarizes which of Piatetsky-Shapiro's principles for quality function behavior are satisfied by each function discussed in this section.

TABLE 3-2 SUMMARY OF KNOWLEDGE QUALITY FUNCTIONS

Knowledge Quality Function	**Piatetsky-Shapiro's Principles for Behavior of Knowledge Quality Functions**		
	Zero when $p(x) = p(x\|y)$	Increases with $p(x\|y)$	Decreases with $p(x)$ or $p(y)$
Entropy	No	No	No
Rule Interest	Yes	Yes	Yes
Φ	Yes	Yes	Yes
J-measure	Yes	No*	Yes
W	No	Yes	No
Certainty	No**	Yes	No

* Satisfies principle over range $p(x|y) > p(x)$.
** Principle is easily enforced by disregarding results where $p(x|y) <= p(x)$.

Entropy does not satisfy any of the proposed principles for the behavior of value functions, while Piatetsky-Shapiro's Rule Interest and Φ functions satisfy all three. Smyth and Goodman's J-measure satisfies two of the principles; and all three over the range $p(x|y) > p(x)$. Over the full range of $p(x|y)$, Certainty satisfies only one of the principles, and two over the range $p(x|y) > p(x)$.

E. SUMMARY

Data-mining systems are based on inductive reasoning. In this way, useful and previously unknown patterns can be brought to our attention that systems based in deductive reasoning would never discover. Knowledge discovered by data-mining systems must reflect the nature of the question that the user is attempting to answer. To facilitate this, a quality function for knowledge can be developed to reflect desirable and quantifiable characteristics. Enforcement of Piatetsky-Shapiro's principles insures that discovered knowledge has positive information content and is biased towards predictive value and rareness.

IV. TESTING QUALITY FUNCTIONS IN A DATA-MINING SYSTEM

A. NAVAL POSTGRADUATE SCHOOL GENETIC PROGRAM (NPSGP)

The Naval Postgraduate School Genetic Program (NPSGP) is an adaptation of the Simple Genetic Program in C (SGPC) written by Tackett and Carmi. SGPC is available via anonymous ftp from the Santa Fe Institute and the University of Texas and is based on the original LISP code published by Koza in his book, *Genetic Programming*. NPSGP modifies SGPC by adding code that implements data mining. Due to the memory requirements and CPU-intensive nature of this application, the suggested computing platform for this application is a SUN SPARC-10 UNIX workstation.

1. Preparing to Data Mine with NPSGP

Several preparatory steps are necessary before NPSGP can be compiled and executed. First, if not already in that format, the database in question must be converted to a tab-delimited ASCII file. NPSGP performs best if:

- The target attribute is represented by non-continuous (discrete) data.
- The non-target attributes include some represented by discrete classes and some represented by continuous data.
- All continuous data representing attributes are scaled to the same range. (Appendix A describes this process.)

The second step is to put the knowledge quality function into NPSGP terms. This process is described in the next section. The third step is to modify the parameters used by the system (in the default.in file). In this way, the user can influence:

- the maximum number of attributes that can be included in a rule
- the size of the initial population
- growth and selection methods
- how many rules will be printed from each generation.

The maximum number of attributes that can be included in a rule and the size of the initial population may need to be adjusted downward if the memory of the workstation is overwhelmed. The rules are printed in order of decreasing quality. The reader is referred to Koza's book, *Genetic Programming,* for details of growth and selection methods. Appendix B explains these steps in detail.

2. Writing Quality Functions in NPSGP Terms

Implementing a knowledge quality function in NPSGP requires translating it into a "fitness function" in C code. Appendix C includes the fitness functions used in this research. The first step is to identify the fitness function terms equivalent to the probability-oriented terms defined in Chapter III. For each rule evaluated, NPSGP partitions the examples of a database into one of four possible conditions as shown in Table 4-1. The terms used to express the components of the

TABLE 4-1. CONTINGENCY TABLE FOR EACH RULE

	Target Attribute (RHS) is:	
	True	False
Description (LHS) is True	a	b
Description (LHS) is False	c	d

quality functions in the last chapter can now be expressed in the terms of Table 4-1. Quality function and fitness equivalent expressions are shown in Table 4-2.

TABLE 4-2. QUALITY FUNCTION TERMS RESTATED IN FITNESS FUNCTION TERMS

Terms used in Quality Functions	**Description**	**NPSGP Fitness Function Terms**
y	the description of target attribute class *x* in terms of the non-target attributes, or the left hand side (LHS) of a rule.	a + b
x	specific class within *X* that is to be described by *y*, or the right hand side (RHS) of a rule.	a + c
n	total number of examples in the source database.	examples
p(y)	probability of the description (LHS) occurring in the database, sometimes used as a stand-in for description simplicity.	$\frac{a + b}{\text{examples}}$
p(x)	probability that an example is in the class *x*, or the *a priori* value of any rule.	$\frac{a + c}{\text{examples}}$
p(x|y)	probability that *x* will occur given description *y*, or the *a posteriori* value of the rule. This is also known as the conditional probability.	$\frac{a}{a + b}$

After the quality functions have been replaced with the equivalent fitness function terms, the function must be checked for behavior that would cause errors, such as taking the logarithm of zero or a negative number. If necessary, the fitness function is modified or additional programming statements are added to avoid adverse program behavior. Finally, the fitness function must be expressed so that it approaches zero when optimal, because NPSGP optimizes by minimizing the fitness function. Table 4-3 shows the fitness

functions for the three quality functions selected for testing. Some additional code was added so that only rules with positive information gain would participate in the crossover and mutation operations when the J-measure and Certainty functions were used.

TABLE 4-3. QUALITY FUNCTIONS REPRESENTED IN FITNESS FUNCTION TERMS

Quality Function	Representation in Fitness Function Terms
Rule Interest	1 - [(a/examples) - [(a+c)/examples * (a+b)/examples]]
J-measure	1 - (((a+b)/examples) * ((a/(a+b)) * (log10(a/(a+b)) - log10((a+c)/examples)) + ((1-(a/(a+b))) * (log10(1.001-(a/(a+b))) - log10(1-((a+c)/examples))))))
Certainty	1 - [a/(a + b)], a+b > 500

B. DESCRIPTION OF MUSHROOM DATABASE

The experiments are performed upon a database obtained from the University of California at Irvine's Repository of Machine Learning Databases and Domain Theories and is available via anonymous ftp at ics.uci.edu:pub/machine-learning-databases. Schlimmer donated the mushroom data set, whose attributes include descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the genera *Agaricus* and *Lepiota* (Schlimmer, 1987). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. Schlimmer combined the latter class with the poisonous one. The *Audubon Society Field Guide to North American Mushrooms* clearly states that there is no simple rule for determining the edibility of a mushroom (Lincoff, 1981, p. 871).

1. Size and Attributes

The mushroom database consists of 8,124 examples. In general, the attributes in the database deal with the characteristics of mushrooms and their edibility. Table 4-4 shows the 23 attributes with their possible discrete states.

2. Necessary Modifications to the Database

Several modifications were needed to the mushroom database to allow the prototype NPSGP data-mining system to function. First, the discrete data of two attributes were reformatted with continuous data. As mentioned in the previous section, NPSGP requires non-target attributes to include some represented by discrete classes and some represented by continuous data. With the mushroom dataset, it was essential to represent at least two non-target attributes with continuous data. The authors speculate that some continuous data are necessary to insure that some randomly generated descriptions in the initial population match the examples in the database. Gill Spacing and Ring Number were chosen for reformatting, as these attributes lend themselves well to continuous descriptors. Second, Gill Spacing and Ring Number were normalized to range between 0 and 100. NPSGP works best if all continuous data are normalized to a uniform range. Appendix A describes the reformatting process.

TABLE 4-4. MUSHROOM DATABASE ATTRIBUTES WITH DISCRETE STATES

Attributes	Possible States
Classes	Edible, Poisonous, Z
Cap Shape	Bell, Conical, Convex, Flat, Knobbed, Sunken
Cap Surface	Fibrous, Grooved, Scaly, Smooth
Cap Color	Brown, Buff, Cinnamon, Gray, Green, Pink, Purple, Red, White, Yellow
Bruises?	True, False
Odor	Almond, Anise, Creosote, Foul, Musty, Pungent, None
Gill Attachment	Attached, Descending, Free, Notched
Gill Spacing	Close (0), Crowded (50), Distant (100)
Gill Size	Broad, Narrow
Gill Color	Black, Brown, Buff, Chocolate, Gray, Green, Orange, Pink, Purple, Red, White, Yellow
Stalk Shape	Enlarging, Tapering
Stalk Root	Bulbous, Club, Cup, Equal, Rhizomorphs, Rooted, Missing
Stalk Surface Above Ring	Fibrous, Scaly, Silky, Smooth
Stalk Surface Below Ring	Fibrous, Scaly, Silky, Smooth
Stalk Color Above Ring	Brown, Buff, Cinnamon, Gray, Orange, Pink, Red
Stalk Color Below Ring	Brown, Buff, Cinnamon, Gray, Orange, Pink, Red, White, Yellow
Veil Type	Partial, Universal
Veil Color	Brown, Orange, White, Yellow
Ring Number	None (0), One (50), Two (100)
Ring Type	Cobwebby, Evanescent, Flaring, Large, Pendant, Sheathing, Zone, None
Spore Print Color	Black, Brown, Buff, Chocolate, Green, Orange, Purple, White, Yellow
Population	Abundant, Clustered, Numerous, Scattered, Several, Solitary
Habitat	Grasses, Leaves, Meadows, Paths, Urban, Waste, Woods

3. Rareness of Target Attribute

We could not test for rare outcomes without changing the database because the target attribute in the mushroom database, "Classes" has only two almost equally distributed states, "Edible" and "Poisonous." To test the ability of the knowledge quality functions to find rare events (*i.e.*, states of the target attribute that occur less often than most), it was necessary to add a new "Rare" state and insure that a good pattern had that state as the outcome. One high quality rule (*i.e.*, IF Odor = None, Then mushroom is Edible) was observed in the database. The examples supporting that rule were found and 25% (852 examples) were changed from state "edible" to state "z." Table 4-5 represents the occurrence of the states in the target attribute before and after adding the new class "z".

TABLE 4-5. RARENESS OF TARGET CLASS BEFORE AND AFTER CHANGING THE DATA SET

Without Rare Rule			**With Rare Rule**		
edible	**poisonous**	**z**	**edible**	**poisonous**	**z**
4,208	3,916	0	3,356	3,916	852
51.8%	48.2%	0%	41.3%	48.2%	10.5%

C. NPSGP OUTPUT

Figure 4-1 shows how NPSGP presents a rule discovered in the mushroom database using Rule Interest as the fitness function.

```
Top 50 of Generation 49
Tree #1
(IF
 (AND
  (AND
   (NOT
    (TWIXT
     -263.782806
     RING_NUMBER
     -222.571823))
   (NOT
    (TWIXT
     -0.231361
     GILL_SPACING
     29.317902)))
  (NOT
   (IN-CATEGORY
    RING_TYPE
    Evanescent )))
 (IN-CATEGORY
  CLASSIFY
  Poisonous ))
Number of records matched by LHS: 3332.000000
Number of misclassified records: 240.000000
Confidence: 0.927971
Validation Fitness= 0.817100
```

Figure 4-1. A Sample Genetic Programming Rule

As can be seen, even in production rule format, the apparent complexity of the rule requires some post-processing before the rule is easily understandable. The rule displayed in Figure 4-1, while appearing complicated is actually:

IF GILL-SPACE is NOT between -0.23 and 29.32 and RING-TYPE is NOT Evanescent, THEN mushroom is Poisonous.

A worksheet was used to facilitate the collection and comparison of the rules discovered by NPSGP. The rule in Figure 4-1 is shown in a rule tracking worksheet in Appendix D.

As mentioned in Chapter III, rule simplicity is considered highly desirable, as simple rules are considered more likely to be correct for new data not represented in the database. Overly complex rules are expected to be "overfitted" and unlikely to stand up in the face of additional examples. Rule simplicity can be enforced within the program or through the fitness function. An automated pruning mechanism similar to those used by some data-mining systems to prevent overfitting would reduce the human task of interpreting the rules. However, any automatic simplification of the rules within the genetic programming process would be extremely undesirable because this simplification would diminish the diversity of the "genetic material" available for crossover operations and so diminish the opportunity to escape from local optima. Therefore, manipulation of the fitness function is the best way to bias the system toward discovery of functionally simple rules.

V. EVALUATION OF KNOWLEDGE QUALITY FUNCTIONS

NPSGP was used to test the performance of Rule Interest, J-measure, and Certainty as fitness functions. If each of these knowledge quality functions discovers a distinctively different set of rules, it becomes possible to choose the most appropriate of them for different data-mining applications. Worksheets showing the best rules discovered using the three functions are in Appendix D. Table 5-1 provides some summary statistics about the rules discovered by the three knowledge quality functions.

TABLE 5-1. SUMMARY STATISTICS ABOUT THE "BEST" RULES

	Knowledge Quality Functions		
	Rule Interest	**J-measure**	**Certainty**
	20 best rules		43 exact rules
Average coverage [1]	41.8%	30.3%	15.3%
Minimum number of examples covered by a description	26.6%	10.6%	7.1%
Maximum number of examples covered by a description	54.3%	51.2%	26.6%
Minimum confidence: $p(x\|y)$	67.4 %	71.8 %	100 %
Maximum confidence: $p(x\|y)$	100 %	100 %	100 %

[1] Average coverage is calculated by summing the number of examples matched by each rule description in a set, dividing by the number of rules, and expressing the result as a percentage.

If the number of attributes included in a rule is considered a measure of complexity, Table 5-2 summarizes the complexity of the "best" rules for each of the knowledge quality functions.

TABLE 5-2. NUMBER OF ATTRIBUTES USED IN RULES

Number of Attributes Used in Rules	Knowledge Quality Functions		
	Rule Interest	**J-measure**	**Certainty**
	20 best rules		43 exact rules
1	20%	50%	26%
2	40%	35%	67%
3	40%	10%	5%
4	0%	5%	2%

To the extent the differences between the sets of rules discovered can be defined, a choice can be made between them as appropriate knowledge quality functions for future data-mining applications.

A. CERTAINTY

The Certainty function finds a wealth of rules, many of which are exact rules: *e.g.*, rules that predict the outcome without any misclassifications. All exact rules are equally valued by the Certainty function. Unconstrained, Certainty discovered a very large number of exact rules, including some that applied to only two or three of the 8124 examples in the Mushroom database. In all, 43 exact rules that applied to at least 500 examples were considered the "best" rules found by Certainty. In general, exact rules that apply to very large proportions of the database are not particularly interesting because they are usually quite obvious. Certainty did not find the rule that was introduced in the database to test for the ability to find "rare" outcomes because its Certainty value was only 25%. The lowest certainty value of the rules output by the program was 70.8%. Only the

best 100 rules were output from each generation. Presumably, if enough rules had been output, the planted rule would have appeared.

Certainty is a useful knowledge quality function when the rules discovered do not apply to either a preponderance or a very small fraction of the examples in the database. The rules then discovered will be neither obvious nor trivial.

B. RULE INTEREST AND J-MEASURE

Both Rule Interest and J-measure knowledge quality functions are based on information theory concepts and attempt to discover rules with essentially the same characteristics. Piatetsky-Shapiro holds that knowledge quality functions which satisfy his three proposed principles will produce the same set of rules. Smyth and Goodman counter that the failure to use log functions will undervalue "rare events" and thus result in different sets of rules (Smyth and Goodman, 1991, p. 163). As expected, the sets of "best" rules discovered by the functions based on information theory, Rule Interest and J-measure, are similar in most respects. A few exact rules that apply to most of the examples in the database were found by both of these knowledge quality functions, but the other biases built into these functions led them to find other, more interesting rules. Both functions found the rule that was planted to test for the ability to find "rare" outcomes, while Certainty did not find this rule. Some differences were also found between the sets of rules discovered by Rule Interest and J-measure. As displayed in Table 5-1, Rule Interest and J-measure differ primarily in the average proportion of the database covered by the respective sets of rules. The descriptions in the rules discovered by Rule Interest all apply to at least a quarter of the database. The descriptions in the rules discovered by

J-measure all apply to at least a tenth of the database. Both functions found rules with descriptions that apply to approximately half of the examples in the database. In general, J-measure is able to discover a greater range of rules, especially those with descriptions that apply to a small proportion of the examples in the database. This capacity would make J-measure a more appropriate knowledge quality function for applications where unusual activity is an important part of the model to be developed; *e.g.*, modeling adversary aircraft behaviors. Rule Interest would be valuable where more general patterns are of interest, such as demand for material stocked in DoD's supply system. The principle difference between the two functions revealed by Table 5-2 is the complexity of the rules discovered: J-measure tends to favor very simple rules with only one attribute in the description, while the bias of Rule Interest towards simple rules is less clear. The difference between the complexity of rules discovered by the two functions does not seem large enough to suggest either Rule Interest or J-measure should be preferred based on this criterion. Both functions generate interesting rules, many of which overlap. If time allows, both functions might be used to develop a more complete model of a database than either alone provides.

VI. CONCLUSIONS AND RECOMMENDATIONS

Performance in many areas of importance to the Department of Defense can be improved if the underlying patterns of behavior are known. The patterns are there, hidden in DoD's many large databases. Computer systems using inductive reasoning are essential to discover this knowledge. This thesis investigates the manner in which data-mining systems discover useful, interesting but currently unavailable knowledge. A data-mining system creates descriptions, evaluates them for usefulness, modifies them, and reevaluates them in an iterative process. The set of rules produced by the data-mining system constitutes a model of the data from which it was derived. Due to the potentially enormous number of rules, the search and evaluation process is the key task of a data-mining system.

A. CONCLUSIONS

Three knowledge quality functions, Certainty, Rule Interest, and J-measure, were evaluated. Each has strengths and weaknesses. Certainty is the most intuitively obvious to the user. Certainty is most likely to find rules that are obvious and in that way reassure the potential user that the data-mining system is capable of finding familiar, general patterns. These patterns however, by our definition, are not interesting if they do not add to the user's base of knowledge. Therefore, other knowledge quality functions should be used by the data-mining system if new, previously unknown knowledge is to be discovered.

Rule Interest and J-measure are successful at discovering new and interesting knowledge. Both are based on information theory, so there is a large overlap in the characteristics of the rules they find. Both functions find exact rules with wide applicability. Both functions find rules with rare outcomes and high information gain. J-measure discovers rules that apply to a smaller proportion of the examples in the database than does Rule Interest. J-measure also tends to discover simpler rules than Rule Interest.

Choice of a knowledge quality function for a data-mining system should be based on the application. If the purpose of data-mining is to discover rules about rare events such as adversary submarine behavior, J-measure would be the preferred knowledge quality function. Rule Interest would be the preferred function for applications requiring knowledge about general but not exact patterns of behavior. Such patterns are useful to support procurement and stocking material held in the military's supply system, for example. Certainty would be the most appropriate knowledge quality function if knowledge about alternative payoffs is desired; for example, this type of knowledge can be used to support decisions about utilization of resources such as medical treatments and education.

B. RECOMMENDATIONS FOR FURTHER RESEARCH

Several areas are candidates for further research based on this study. First, the knowledge quality functions should be validated against actual DoD databases from several functional fields. The databases tested should be of increasing size, both number of examples and number of attributes, eventually testing databases of terabyte size with

millions of examples and hundreds of attributes. This line of research would provide realistic insights about the actual value of data mining to DoD. Second, additional research is needed in the area of termination criteria. This is a well-known problem in the knowledge discovery field. Possible termination criteria are:

- Terminate when a time limit is reached.
- Terminate after a certain number of iterations is completed.
- Terminate when the increase in the total value of the rules discovered drops below a given threshold.

A third potential area of research is to use NPSGP to discover better knowledge quality functions.

APPENDIX A: NORMALIZING CONTINUOUS DATA

NPSGP will typically work only as well as the quality of the attributes from which it is trying to generate a rule. If continuous attribute extremes vary widely in the search space, e.g., 10 and 10,0000, the random numbers generated by NPSGP may not be able to converge. A method of normalizing continuous attributes over a uniform range is described in Table A-1.

TABLE A-1 DATA NORMALIZATION

	1	2	3	4	5	6
A	Original Data			Normalized Data		
B	Range Low	1	1	0	Range Low	0
C	Range High	1000	20	2	Range High	100
D			50	5		
E			100	10		
F			500	50		
G			1000	100		

LOTUS 123 Equations for Normalized Data:
Cell B4 ((B3-$B$2)/$C$2)*$C$6
Cell C4 ((C3-$B$2)/$C$2)*$C$6
Cell D4 ((D3-$B$2)/$C$2)*$C$6 etc.

APPENDIX B: NPSGP USERS MANUAL

A. NPSGP PREPARATION

Default settings may be made using a UNIX command line editor "Vi", "emacs", etc. to modify the NPSGP source code.

1. default.in

Default configuration of the default random seed, checkpoint_frequency, population size, number of rules reported, etc. Figure B-1 illustrates default.in.

```
seed = 579482
checkpoint_frequency = 1
population_size = 2000
max_depth_for_new_trees = 6
max_depth_after_crossover = 12
max_mutant_depth = 4
grow_method = RAMPED
selection_method = FITNESSPROP
tournament_K = 6
crossover_func_pt_fraction = 0.2
crossover_any_pt_fraction = 0.2
fitness_prop_repro_fraction = 0.1
parsimony_factor = 0.00000
number_reported = 50
```

Figure B-1. Sample NPSGP Default File

2. **setup.c**

The last line of source code in setup.c provides assignment of the initial random constants for floating point generation. The range of this floating point is critical in generation of continuous variables in the initial population. Figure B-2 illustrates the line of code and editing points to establish this constant.

```
{ return ((GENERIC)random_float(10.0) - GENERIC)0.0);
```

Figure B-2. Initial Random Constant Floating Point

3. **fitness.c .**

Two sections of fitness.c require modification prior to NPSGP execution. They are number of examples/tuples in database and Quality Function Selection.

Fitness.c also contains additional code to force NPSGP to produce rules under bounds established by the authors. Comment out these penalty functions if unconstrained program is desired. A listing of fitness.c with explanation of the additonal code is found in Appendix C.

B. COMPILING NPSGP

The TAB delimited ASCII file must be converted to a C-structure for use by NPSGP. Procedure as follows:

1. ***perl define.pl cars1.tab,*** **Converts the data file.**
2. ***make,*** **Compiles the NPSGP source code.**

C. EXECUTION OF NPSGP

Execute NPSGP with command, *gpc 1 50 none 4 > test.out&.*

1. ***nice*** ———— **gives secondary priorty to gpc program**
2. ***gpc*** ———— **execute command**
3. ***1*** ———— **specifies the number of populations**
4. ***50*** ———— **number of generations to run**
5. ***none*** ———— **default.in file**
6. ***4*** ———— **seed number**
7. ***file_name.out*** — **redirects the output to file_name.out**
8. ***&*** ———— **runs the program in background**

D. TRACKING NPSGP OUTPUT

1. Documentation of runs.

Documentation of each run is a necessity when running multiple runs with multiple fitness function on multiple databases. Table B-1 illustrates a run tracking sheet.

2. Visualize output.

The UNIX command **tail -f** *filename* directs gpc output to the screen. This allows the user to visualize and verify the output of the program.

TABLE B-1. NPSGP RUN TRACKING SHEET

1	**Database Name**			
2	Date/Time Start:			
3	UNIX PID (command: ps -lax)			
4	Date/Time Finish:			
5				
6	**Filename.out**			
7	Database ---------- mu_			
8	Fitness Function -- jm_			
9	Date ---------- 730			
10	UNIX Machine ---- in210_			
11	***** To Start GP *****			
12				
13	**gpc 1 50 none > filename.out&**			
14	**tail -f filename.out**			
15				
16	***** Before You Start *****			
17	Name database = cars1.tab			
18	perl define.pl cars1.tab			
19	make			
20				
21	fitness.c			
22	tuples =			
23	fitness function =			
24				
25	structure.c			
26	Change RHS_V = to Problem Field			
27				
28	default.in			
29	population size			
30	number of rules			
31	setup.c			
32	random_ float(10.0) - (generic)0.0);			

APPENDIX C: FITNESS FUNCTIONS

```
/* a+b = number of total matches
   c+d = number of total mismatches
   a+c = number in category
   b+d = number out of category
*/
if(i >= 0)
 lasti = i;
totmatch += a+b;
 if (a == 0)
  result = 10000000;
 else

 if (a/(a+b) <= (a+c)/tuples)
   result = 10000000;
 else

 if ((a+b) <= 500)
   result = 10000000;
 else
```

Prevents consideration of rules where RHS = LHS is always false.

Prevents consideration of rules without positive information gain.

Prevents consideration of rules that apply to less than 501 examples/tuples.

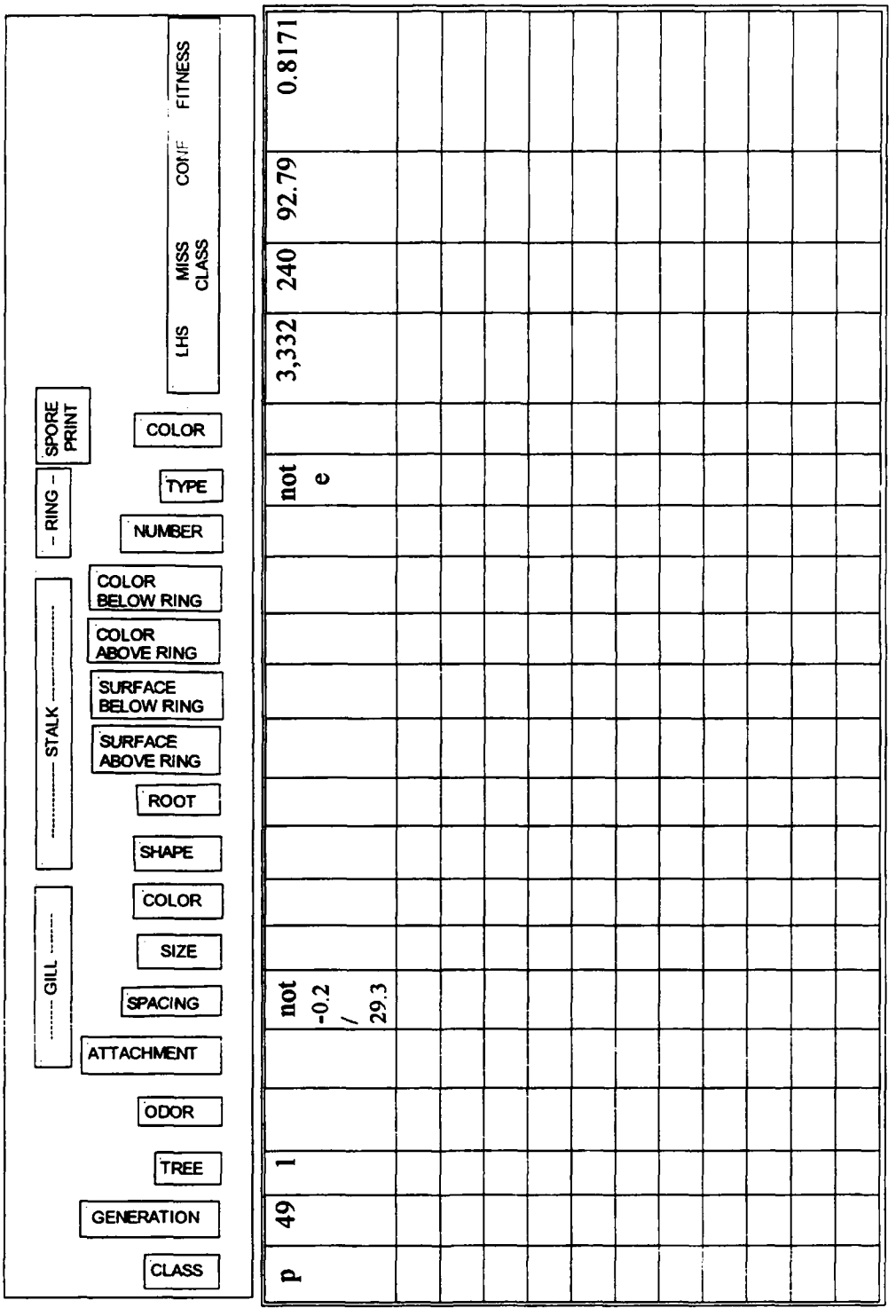
```
/* J-measure */
/*  {  result = 1 - (((a+b)/tuples) * ((a/(a+b)) * (log10(a/(a+b)) - log10((a+c)/tuples)) +
((1-(a/(a+b))) * (log10(1.001-(a/(a+b))) - log10(1-((a+c)/tuples))))));
        }  */

/* Rule Interest  */
/*  result = 1-(((a/(a+b))-((a+c)/tuples))*((a+b)/tuples)); */
/* Confidence*/
   result = 1-(a/(a+b)-.001);

 return(result  );
```

Quality functions are commented out if not to be used, e.g. /* comment*/.

Confidence quality function is selected for this run.

APPENDIX D: RULE TRACKING SHEETS

Sample Rule Tracking Sheet



RULE INTEREST. SUMMARY OF 20 BEST RULES

			----- CAP -----					------- GILL -------				---------- STALK ----------							-- VEIL --		-- RING								
CLASS	GENERATION	TREE	SHAPE	SURFACE	COLOR	BRUISES	ODOR	ATTACHMENT	SPACING	SIZE	COLOR	SHAPE	ROOT	SURFACE ABOVE RING	SURFACE BELOW RING	COLOR ABOVE RING	COLOR BELOW RING	TYPE	COLOR	NUMBER	TYPE	SPORE COLOR	POPULATION	HABITAT	LHS	MISS CLASS	CONF	FITNESS
p	49	1							-0.2/29.3												not e				3332	240	92.79	0.8171
p	2	1				not t			0/3																3540	352	90.0	0.817624
p	47	1							-0.2/29.3											-263/7.5	not e				3296	240	92.7	0.819395
p	49	2												not s						23.0 84.	not e				4072	864	78.7	0.846728
p	43	1							9.8/190											15.1 78.5	not e				4128	960	76.7	0.854975
p	23	1													s					7 64	not e				4264	1056	75.2	0.85812
p	18	1							10/908						not s							not n			2532	160	93.6	0.858259
p	3	2				f														7 55					4408	1152	73.8	0.860756
p	0	1					f																		2160	0	100.0	0.862282
e	13	1								b					s										3400	889	73.8	0.863802
p	13	2							10 908						not s										2628	256	90.2	0.863955
e	1	6					n																		3528	972	72.4	0.864772
p	8	2																			not e				4156	1056	74.5	0.865006
p	12	3							-0.4/4						not s					10 908					2592	256	90.1	0.86625
p	0	3												k											2372	144	93.9	0.866491
p	30	6																		15.1 959	not e				4120	1056	74.3	0.867302
p	0	7													k										2304	144	93.7	0.870826
e	30	7								not n											e				3320	913	72.5	0.872536
p	28	4							-4.0/9.3	n										17.7 672.2					2256	144	93.6	0.873887
e	36	17												s								not r			3912	1273	67.4	0.874081

J-MEASURE. SUMMARY OF 20 BEST RULES

			----- CAP -----						------- GILL -------				------------- STALK						-- VEIL --		-- RING --								
CLASS	GENERATION	TREE	SHAPE	SURFACE	COLOR	BRUISES	ODOR	ATTACHMENT	SPACING	SIZE	COLOR	SHAPE	ROOT	SURFACE ABOVE RING	SURFACE BELOW RING	COLOR ABOVE RING	COLOR BELOW RING	TYPE	COLOR	NUMBER	TYPE	SPORE COLOR	POPULATION	HABITAT	LHS	MISS CLASS	CONF	FITNESS
p	9	1							0/4.4					k						10.7/1538					2192	0	100	0.914487
p	0	1					f																		2160	0	100	0.915736
p	18	1				not t			0.2/4.4											10.7/1180.6					3504	352	89.9	0.925553
p	0	3									b														1728	0	100	0.932589
p	10	3									b									0/0.7					1728	0	100	0.932589
p	0	5												k											2372	144	94	0.93692
p	0	8													k										2304	144	94	0.939345
p	0	12																				w			1632	48	97	0.949009
p	0	13																			f				1296	0	100	0.949441
p	0	17								n															2512	288	89	0.950801
p	0	20								not b															2512	288	89	0.950801
e	5	34					n																		3528	972	72	0.962345
p	0	28												not s											2948	568	81	0.964255
p	0	29								n				k											896	0	100	0.965046
p	23	34		not f					3.3/667						not s					2.8/203					1928	256	86.7	0.966033
p	2	40								n							p								864	0	100	0.966294
e	8	38				t	not p																		3120	856	72.5	0.966446
e	9	32				t							not e												3120	856	72.5	0.966446
e	30	56									not b			s											4312	1386	67.8	0.966748
p	7	33																			not e				4156	1056	74.5	0.967631

CERTAINTY: SUMMARY OF 43 EXACT RULES

			CAP					GILL				STALK						VEIL		RING								
CLASS	GENERATION	TREE	SHAPE	SURFACE	COLOR	BRUISES	ODOR	ATTACHMENT	SPACING	SIZE	COLOR	SHAPE	ROOT	SURFACE ABOVE RING	SURFACE BELOW RING	COLOR ABOVE RING	COLOR BELOW RING	TYPE	COLOR	NUMBER	TYPE	SPORE COLOR	POPULATION	HABITAT	LHS	MISS CLASS	CONF.	FITNES
p	0	1					f																		2160	0	100.0	0.001
p	0	2			y	f																			672	0	100.0	0.001
p	0	3																f							1296	0	100.0	0.001
p	0	4									b														1728	0	100.0	0.001
p	0	6					a																		576	0	100.0	0.001
p	0	8											b											m	612	0	100	0.001
p	2	1					y																		576	0	100.0	0.001
p	3	32				not t	not n																		3252	0	100.0	0.001
p	4	34								n	b														1728	0	100.0	0.001
p	5	11		y										k											1132	0	100.0	0.001
p	5	46			a	f																			876	0	100.0	0.001
p	8	18													k						-3.4 5	4.2			2160	0	100.0	0.001
p	16	26								not n													w		1584	0	100.0	0.001
p	16	26				f	a																		576	0	100.0	0.001
p	19	27				f	f																		1872	0	100.0	0.001
p	19	40								b					k										864	0	100.0	0.001
p	19	41				f																f			1296	0	100.0	0.001
p	19	44					a														-9.8/9.				576	0	100.0	0.001
p	20	46							4.9 62.5						k								w		1296	0	100.0	0.001
p	21	23									b											not e			1728	0	100.0	0.001
p	21	38					a							f											576	0	100.0	0.001
p	22	10													k										864	0	100.0	0.001
p	22	21					a																		576	0	100.0	0.001
p	22	22							7.0/43.5						k										2160	0	100.0	0.001
p	23	9								n not b					k										864	0	100.0	0.001
p	23	12			e					n															864	0	100.0	0.001
p	25	34					f		-63.8/64.2																2160	0	100.0	0.001
p	26	1								n				k											896	0	100.0	0.001
p	26	14					f														-7.9/6.				2160	0	100.0	0.001
p	28	11					a															not e			576	0	100.0	0.001
p	28	45																				f not e			1296	0	100.0	0.001
p	29	41					a																		576	0	100.0	0.001
p	30	2					f			not n															1584	0	100.0	0.001
p	30	6					not f																		576	0	100.0	0.001
p	30	9								not n												f			1296	0	100.0	0.001
p	30	49							-18.0/7.2		b														1728	0	100.0	0.001
p	32	16																			-1.3/1.	f			1296	0	100.0	0.001
p	33	2					f							not f											2016	0	100.0	0.001
p	33	32																				f not e not n			1296	0	100.0	0.001
p	34	13					y not a																		576	0	100.0	0.001
p	34	36								b				not f	k							not e			1296	0	100.0	0.001
p	36	13									b				not k										764	0	100.0	0.001
p	36	25								n					k							not e			864	0	100.0	0.001

REFERENCES

Agrawal, Rakesh, Tomasz Imielinski, and Arun Swami, "Database Mining: A Performance Perspective," *IEEE Transactions on Knowledge and Data Engineering*, December, 1993.

Chan, Keith C. C. and Wong, Andrew K. C. "A Statistical Technique for Extracting Classificatory Knowledge from Databases," in *Knowledge Discovery in Databases*, eds. Gregory Piatetsky-Shapiro and William J. Frawley, AAAI Press, Menlo Park, CA, 1991.

The Concise Columbia Encyclopedia, Columbia University Press, 1983.

Dhar, Vasant and Alexander Tuzhilin, "Abstract-Driven Pattern Discovery in Databases," *IEEE Transactions on Knowledge and Data Engineering*, December, 1993.

Frawley, William J., Gregory Piatetsky-Shapiro and Christopher J. Matheus, "Knowledge Discovery in Databases: An Overview," in *Knowledge Discovery in Databases*, eds. Gregory Piatetsky-Shapiro and William J. Frawley, AAAI Press, Menlo Park, CA, 1991.

Frawley, William J., "Using functions to Encode Domain and Contextual Knowledge in Statistical Induction", in *Knowledge Discovery in Databases*, eds. Gregory Piatetsky-Shapiro and William J. Frawley, AAAI Press, Menlo Park, CA, 1991.

Goldberg, David. E., "Genetic and Evolutionary Algorythms Come of Age", in *Communicaitons of the ACM*, March 1994.

Goldberg, David. E., "Genetic Algorithms and Rule Learning in Dynamic System Control," in , *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, 1985.

Grefenstette, John J., "Genetic Algorithms," in *IEEE Expert*, ed. B. Chandrasekaran, Los Alamitos, CA, October,1993.

Holsheimer, Marcel and Arno Siebes, *Data Mining: The Search for Knowledge in Databases*, Report CS-R9406, CWI, Amsterdam, The Netherlands, 1994.

Inmon, William H., "Untangling the Web," in *Database Programming and Design*, May, 1993.

Koza, John R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, Massachusetts, 1992.

Levy, Steven, *Artificial Life*, Vintage Books, New York, 1992.

Lincoff, Gary H., *National Audubon Society: Field Guide to North American Mushrooms*, Alfred A. Knopf,Inc., New York, 1981.

Major, John A. and Mantgano, John J., "Selecting Among Rules Induced from a Hurricane Database", in *Knowledge Discovery in Databases: Papers from the 1993 Workshop Technical Report*, AAAI Press, Menlo Park, CA, 1993.

Pazzani, Michael, Patrick Murphy, Kamal Ali, and David Schulenburg, "Trading Off Coverage for Accuracy in Forecasts: Application to Clinical Data Analysis," in *Working Notes AAAI Spring Symposium Series*, Stanford University, 1994.

Piatetsky-Shapiro, George, "Discovery, Analysis, and Presentation of Strong Rules," in *Knowledge Discovery in Databases*, eds. Gregory Piatetsky-Shapiro and William J. Frawley, AAAI Press, Menlo Park, CA, 1991.

Piatetsky-Shapiro, George, Christopher J. Matheus, Padhraic Smyth and Ramasamy Uthurusamy, "KDD-93: Progress and Challenges in Knowledge Discovery in Databases," in *Knowledge Discovery in Databases 1993*, eds. Gregory Piatetsky-Shapiro and William J. Frawley, AAAI Press, Menlo Park, CA, November, 1993.

Quinlan, J. Ross, "The Effect of Noise on Concept Learning," in *Machine Learning II*, eds. Michalski, Ryszard S., Jaime G. Carbonell and Tom M. Mitchell, Machine Learning, An Artificial Intelligence Approach, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1986.

Robey, Brian L., , "DRAIR ADVISER: A Knowledge-Based System for Materiel-Deficiency Analysis," *AI Magazine*, Summer 1994.

Schimmler, J.C., *Concept Acquisition through Representational Adjustment*, Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine, 1987.

Smyth, Padhraic and Rodney M. Goodman, "Rule Induction Using Information Theory," in *Knowledge Discovery in Databases*, eds. Gregory Piatetsky-Shapiro and William J. Frawley, AAAI Press, Menlo Park, CA, 1991.

Smyth, Padhraic and Rodney M. Goodman, "An Information Theoretic Approach to Rule Induction from Databases," in *IEEE Transactions on Knowledge and Data Engineering*, August, 1992.

Weiss, Neil A., and Matthew J. Hassett, *Introductory Statistics*, Addison-Wesley, Reading, MA, 1991.

Yasdi, Ramin, "Learning Classification Rules from Database in the Context of Knowledge Acquisition and Representation," in *IEEE Transactions on Knowledge and Data Engineering*, September, 1991.

Ziarko, Wojciech, "The Discovery, Analysis, and Representation of Data Dependencies in Databases," in *Knowledge Discovery in Databases*, eds. Gregory Piatetsky-Shapiro and William J. Frawley, AAAI Press, Menlo Park, CA, 1991.

Zytkow, Jan M., "Introduction: Cognative Autonomy in Machine Discovery", in *Machine Learning*, 12, 1993.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, VA 22304-6145

2. Library, Code 52 — 2
   Naval Postgraduate School
   Monterey, CA 93943-5101

3. Dr. Balasubramanium Ramesh, Code SM/RA — 4
   Department of Systems Management
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. CDR William B. Short, Code SM/WS — 1
   Department of Systems Management
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. LCDR Elizabeth S. Walters — 3
   P. O. Box 492
   Mechanicsburg, PA 17055

6. LCDR Frank J. Bunn — 3
   104 Quail Road
   Madison, AL 35758